

SciToolAgent: A Knowledge Graph-Driven Scientific Agent for Multi-Tool Integration

Keyan Ding^{1,2†}, Jing Yu^{2,4†}, Junjie Huang², Yuchen Yang²,
Qiang Zhang^{3,2*}, HuaJun Chen^{1,2,5*}

¹College of Computer Science and Technology, Zhejiang University,
Hangzhou, China.

²Zhejiang Key Laboratory of Intelligent Manufacturing for Functional
Chemicals, ZJU-Hangzhou Global Scientific and Technological
Innovation Center, Zhejiang University, Hangzhou, China.

³ZJU-UIUC Institute, Zhejiang University, Haining, China.

⁴The Polytechnic Institute, Zhejiang University, Hangzhou, China.

⁵State Key Laboratory of Ocean Sensing, Hangzhou, China.

*Corresponding author(s). E-mail(s): qiang.zhang.cs@zju.edu.cn;
huajunsir@zju.edu.cn;

Contributing authors: dingkeyan@zju.edu.cn; yujing17@zju.edu.cn;
junjie6282@zju.edu.cn; yangyc_2020@163.com;

[†]These authors contributed equally to this work.

Abstract

Scientific research increasingly relies on specialized computational tools, yet effectively utilizing these tools demands substantial domain expertise. While Large Language Models (LLMs) show promise in tool automation, they struggle to seamlessly integrate and orchestrate multiple tools for complex scientific workflows. Here, we present SciToolAgent, an LLM-powered agent that automates hundreds of scientific tools across biology, chemistry, and materials science. At its core, SciToolAgent leverages a scientific tool knowledge graph that enables intelligent tool selection and execution through graph-based retrieval-augmented generation. The agent also incorporates a comprehensive safety-checking module to ensure responsible and ethical tool usage. Extensive evaluations on a curated benchmark demonstrate that SciToolAgent significantly outperforms existing approaches. Case studies in protein engineering, chemical reactivity prediction, chemical synthesis, and metal-organic framework screening further

demonstrate SciToolAgent’s capability to automate complex scientific workflows, making advanced research tools accessible to both experts and non-experts.

1 Introduction

Scientific research increasingly depends on specialized computational tools that automate essential tasks ranging from data analysis to result visualization. While these tools have become indispensable for advancing scientific discovery, their growing complexity and diversity create substantial barriers to adoption. For example, Chemists routinely employ specific tools for molecular simulation, property prediction, and compound design. Beginner researchers may lack the technical expertise required to effectively utilize these powerful resources, potentially impeding scientific progress. To address this challenge, a promising approach involves leveraging artificial intelligence (AI) technologies to efficiently organize and orchestrate the use of scientific tools [1, 2, 3].

Large language models (LLMs), as cutting-edge AI methods, have demonstrated unprecedented capabilities across diverse domains, from natural language understanding to complex reasoning tasks [4, 5, 6]. Recent research has demonstrated promising advances in integrating LLMs with domain-specific scientific tools [7, 8]. In chemistry, several pioneering systems—including Coscientist [9], ChemChat [10], ChemCrow [11], and CACTUS [12]—have enabled autonomous chemical research through LLM-tool integration. Similar progress has emerged in biological sciences, where systems like GeneGPT [13], CRISPR-GPT [14], GenoAgent [15], and ProtAgents [16] have enhanced LLMs for specialized tasks such as RNA sequencing, gene editing, and protein discovery. The application of tool-augmented LLMs has further expanded to materials discovery (LLMatDesign [17], ChatMOF [18]), electronic design (ChatEDA [19]), and mechanics engineering (MechAgents [20]). These systems typically leverage in-context learning within established frameworks like ReAct [21], which combines LLM-based reasoning with tool execution. However, current approaches face two critical limitations: (1) they operate with a restricted set of tools (typically fewer than twenty), limiting their broader applicability, and (2) they often overlook crucial safety and ethical considerations in scientific research [22].

Current agent frameworks that rely on naive in-context learning often struggle with complex scientific problems due to their inability to account for intrinsic dependencies among a wide array of tools [23, 24]. These dependencies are predominantly characterized by sequential relationships, where the output of one tool serves as the input for the next, necessitating a precise operational order. The failure to account for these interdependent relationships frequently leads to suboptimal solutions and reduced efficiency when handling multi-step scientific workflows. In this study, we present SciToolAgent, an agent framework that effectively integrates extensive and diverse scientific tools with LLMs (Fig. 1). Specifically, SciToolAgent utilizes advanced LLMs as Planner, Executor, and Summarizer to autonomously plan, execute multiple tools, and summarize the solution for scientific tasks. SciToolAgent introduces two

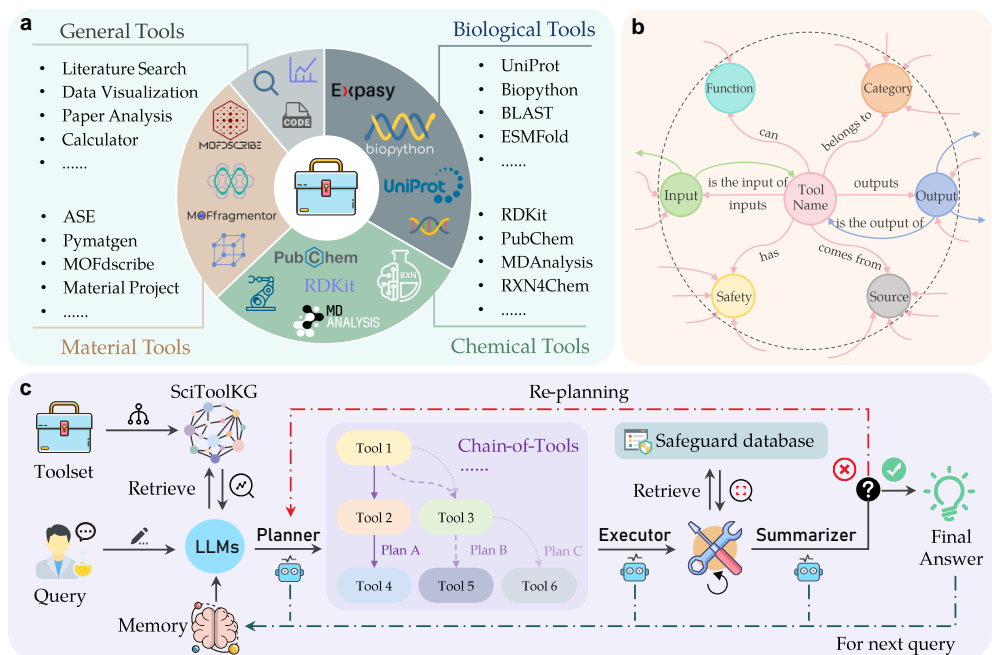


Fig. 1 Overview of SciToolAgent. (a) The toolset involved in our study, which includes general tools as well as frequently-used biology, chemistry, and material-related tools. (b) The schema of SciToolKG, encompassing diverse information about each tool such as input/output formats, specific functionalities, safety levels, etc. (c) The overall workflow of SciToolAgent. Given a user query, an LLM-based Planner generates a chain of tools using a SciToolKG-based retrieve-augmented generation, followed by sequential implementation through an LLM-based Executor. A safety check module is then conducted to ensure ethical and safe solutions by retrieving a safeguard database. Finally, an LLM-based Summarizer concludes the implementation results, assesses the problem-solving process, and prompts the Planner to generate a new chain-of-tools if necessary. The final answers will be sent to the memory module as context for the next query.

key innovations: a comprehensive scientific tool knowledge graph (SciToolKG) that encodes relationships among hundreds of tools across biology, chemistry, and materials science, and an integrated safety module that ensures responsible scientific research. The SciToolKG enables informed tool selection and combination by explicitly modeling tool dependencies, prerequisites, and compatibility. Our safety module continuously monitors the execution pipeline to prevent potentially harmful outcomes, addressing a critical limitation in existing frameworks that often overlook the ethical implications of automated scientific discovery.

We evaluate SciToolAgent using our scientific tool evaluation (SciToolEval) benchmark, comprising 531 diverse scientific problems that span multiple domains and complexity levels. Quantitative analysis demonstrates that SciToolAgent achieves an overall accuracy of 94%, surpassing state-of-the-art baselines by 10%. We further validate SciToolAgent’s effectiveness through case studies across four scenarios: protein

design and analysis, chemical reactivity prediction, chemical synthesis and analysis, and metal-organic framework (MOF) material screening. These studies showcase SciToolAgent’s capability to autonomously orchestrate complex multi-tool workflows while maintaining solution reliability and accuracy.

2 Results

2.1 Overview of SciToolAgent

SciToolAgent is an LLM-based agent designed to overcome the limitations of existing systems in scientific tool integration. While traditional frameworks often fail due to their naive in-context learning approaches that overlook tool interdependencies, SciToolAgent addresses this challenge through a scientific tool knowledge graph (SciToolKG) that mediates between LLMs and scientific tools. The toolset involved in SciToolAgent includes general tools like search engines and code interpreters, as well as specialized scientific tools in biology, chemistry, and materials domains (Fig. 1a). To establish the intricate relationships among extensive tools, we manually construct a SciToolKG (Fig. 1b), encompassing diverse information about each tool. This SciToolKG plays a crucial role in the planning process, enabling LLMs to make well-informed decisions regarding tool selection and combination for optimal problem-solving. The implementation workflow of SciToolAgent is illustrated in Fig. 1c, which consists of three main components: a tool *Planner*, a tool *Executor*, and a solution *Summarizer*, all of which are powered by LLMs. The Planner utilizes retrieve-augmented generation with SciToolKG to generate a chain of tools that can solve the given query. The Executor implements these tools sequentially and retries if errors occur. A retrieve-based safety check module is employed to identify the potential harmful responses within tools. The Summarizer compiles and synthesizes outputs from various tools to generate the final answer. In case of failure to solve the problem using the current plan, as judged by the summarizer, it will prompt the Planner to refine the chain of tools.

2.2 Performance on SciToolEval

Dataset: To quantitatively evaluate the performance of SciToolAgent, we curated SciToolEval, a comprehensive scientific tool evaluation dataset, comprising 531 diverse scientific questions across various fields, such as molecular property prediction, protein analysis, and material retrieval. The process of dataset construction is elaborated in Sec. 4.3. The final dataset is divided into two levels: Level-1 includes 152 questions that can be addressed using a single tool, while Level-2 comprises 379 questions that require the use of multiple tools.

Evaluation metrics: To rigorously assess the effectiveness of SciToolAgent, we utilize three distinct evaluation metrics: (1) Pass Rate, which quantifies the proportion of successfully completed queries; (2) Tool Planning Accuracy, which measures the alignment between the tool selection and sequencing generated by the agent and a reference plan verified by human experts; (3) Final Answer Accuracy, which compares the final solution generated by agents with the standard answer, assessing the

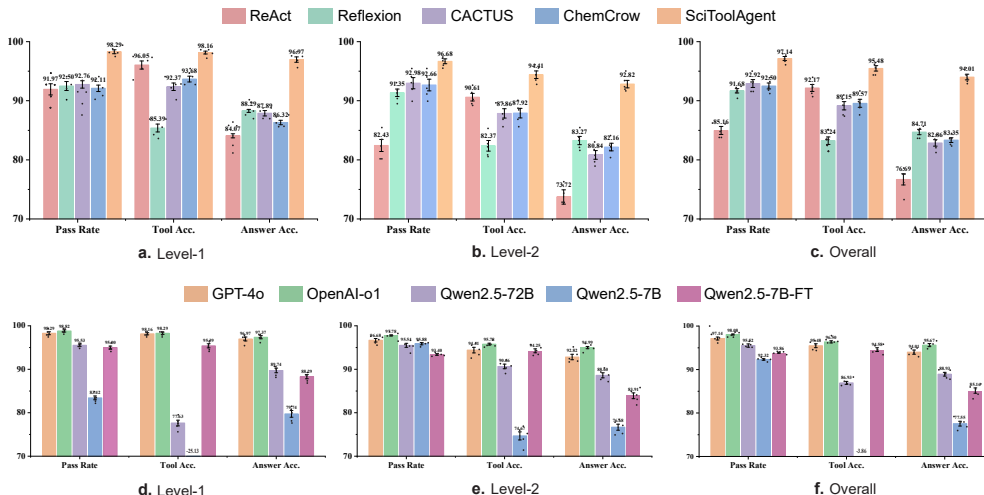


Fig. 2 Comparison results of different agents and foundation models. (a)-(c) Performance of different agents on SciToolEval, including Level-1 and Level-2, as well as overall performance. The proposed SciToolAgent consistently outperforms the baseline approaches across all evaluation metrics and levels. (d)-(f) Performance of different foundation models within SciToolAgent. The OpenAI-o1 model achieves the best result, while the GPT-4o reaches an optimal trade-off between accuracy and cost. Data are presented as mean values. The number of experimental replicates is $n = 5$, and the error bars are standard error of the mean.

correctness of the output. For both tool planning accuracy and final answer accuracy, we prompt GPT-4o to perform a similarity evaluation between the generated response and the ground truth.

Baselines: We compare SciToolAgent’s performance against two state-of-the-art LLM-based tool agents including ReAct [21], Reflexion [25], CACTUS [12], and ChemCrow [11]. The ReAct framework involves an iterative process where LLMs generate a reasoning step (that is, thought) to determine how to tackle the problem and select a tool (that is, action) to take, followed by a result analysis (that is, observation) through LLMs. The Reflexion framework enhances ReAct by incorporating a feedback mechanism into the decision-making process, which includes self-evaluation, self-reflection and memory components. ChemCrow and CACTUS are ReAct-based agents with domain-specific tools and templates for chemical synthesis and material science, respectively. While their default toolsets are limited, we adapt them with our full toolset for a fair comparison.

Foundation models: In our experiments, we utilize different foundation models for the Planner, Executor and Summarizer, including the proprietary model - OpenAI’s GPT-4o (default) and o1, as well as the state-of-the-art open-source model - Qwen2.5-72B. All of these models are accessed via official APIs with a standard configuration. Additionally, we explore Qwen2.5-7B, an open-source LLM with a smaller parameter scale but higher efficiency. Given its reduced inference capacity, we fine-tuned it using specific instructions for tool learning. This setup allows us to evaluate the cost-effectiveness of SciToolAgent across models of varying capacities, ensuring

that our framework can perform effectively while accommodating resource-constrained environments.

Result Analysis: Fig. 2a-c show the performance of five GPT-4o-based agents (ReAct, Reflexion, ChemCrow, CACTUS, and SciToolAgent) across different levels on SciToolEval. SciToolAgent outperformed the other agents in all evaluated metrics. For the task of using a single tool (Level-1), the agent needs to select an appropriate tool from the candidate tools, execute it correctly, and summarize the answer. For the task of using multiple tools (Level-2), the agent requires additional capabilities, including selecting multiple tools, generating a suitable plan, and executing them in a specific sequence. SciToolAgent demonstrates a more significant advantage in solving multi-tool problems, outperforming ReAct and Reflexion by approximately 20% and 10%, respectively, in terms of final answer accuracy. Compared to ReAct-based agents ChemCrow and CACTUS, SciToolAgent also achieves an absolute gain of 10–12% on both levels. Empirically, we observe that ReAct and Reflexion exhibit limitations in tool planning and execution when confronted with complex multi-tool tasks and a large set of candidate tools. Both approaches lack a comprehensive, global task-planning strategy. Although Reflexion employs a feedback mechanism to improve pass rate and accuracy, its extensive trial-and-error process results in a lower tool planning accuracy. SciToolAgent utilizes SciToolKG, significantly enhancing the accuracy of tool planning. Owing to the introduction of the chain-of-tools, SciToolAgent can plan step-by-step tasks from a global perspective, reducing trial-and-error costs and improving its planning proficiency. During tool execution, the chain-of-tools helps the model focus on the current subtask, avoiding interference from irrelevant information, thus enhancing execution accuracy.

Fig. 2d-f show the performance of five different foundation models within the SciToolAgent framework. OpenAI o1 consistently outperforms other models across all metrics in both Level-1 and Level-2 tasks, followed by GPT-4o, which achieves slightly lower accuracy but maintains strong overall performance. Qwen2.5-72B performs competitively but demonstrates a slight drop in tool planning accuracy and final answer accuracy. In contrast, Qwen2.5-7B shows comparatively lower performance, especially in tool planning, indicating limitations in its capacity to manage complex tool integration. However, a finetuned version of Qwen2.5-7B (FT), trained using data generated from SciToolKG, exhibits significant improvement (+10%) and achieves results closer to the higher-capacity models (Qwen2.5-72B). Notably, GPT-4o achieves an optimal trade-off between accuracy and cost, delivering near-top performance across all metrics while maintaining a significantly lower API cost compared to o1. This makes GPT-4o the default foundation model in our implementation of SciToolAgent.

2.3 Case Studies

To further validate the utility of SciToolAgent, we conduct four real-world scientific research tasks with SciToolAgent, including (1) protein design and analysis, (2) chemical reactivity prediction, (3) chemical synthesis and analysis, and (4) MOF materials screening.

2.3.1 Protein design and analysis

Protein design is crucial for advancing biological fields such as drug discovery, enzyme engineering, and synthetic biology. Designing proteins with desired properties is a complex task that typically requires the integration of multiple bioinformatics tools for sequence generation, folding prediction, and structural analysis. Traditional workflows necessitate substantial expertise to navigate and operate the diverse set of specialized tools effectively. SciToolAgent streamlines this process by autonomously orchestrating the necessary tools to achieve comprehensive protein design and analysis.

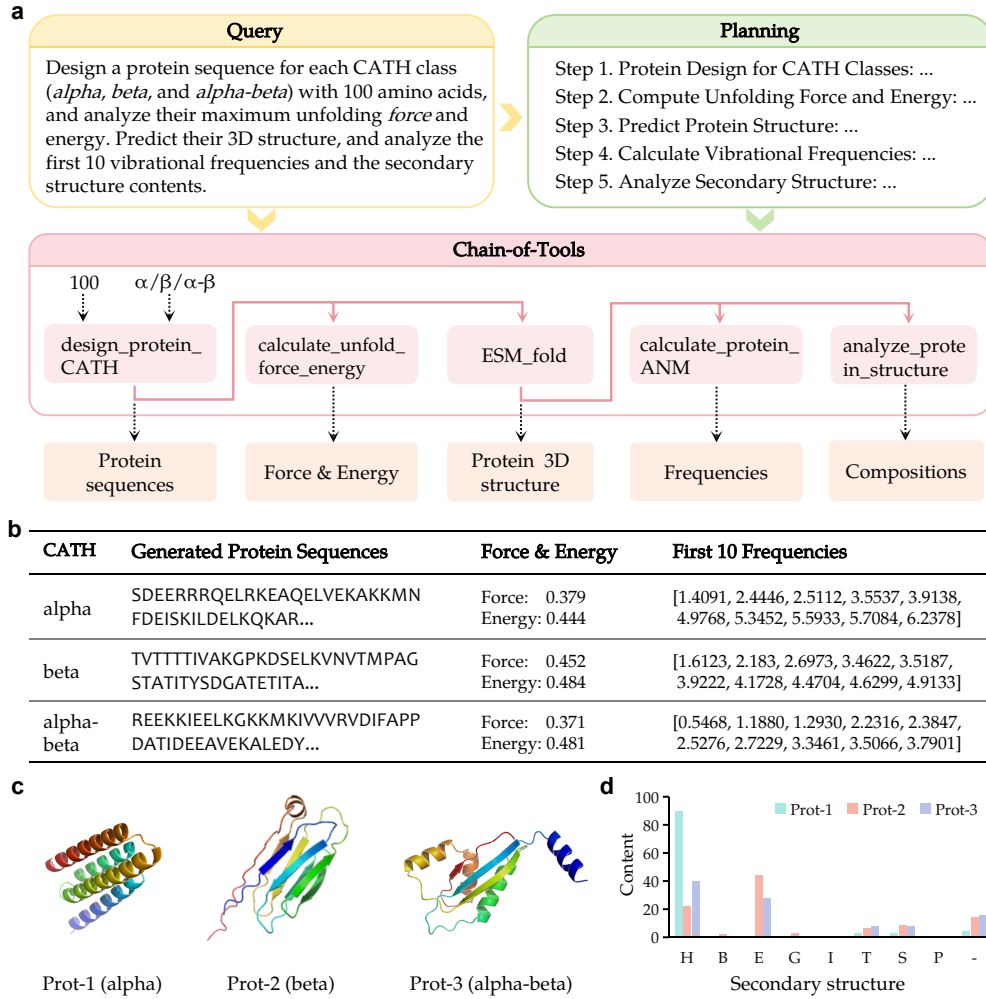


Fig. 3 Workflow and results of protein design and analysis using SciToolAgent. (a) Overview of the input query, tool planning, chain-of-tools, and corresponding outcomes. (b) The generated protein sequences, the corresponding unfolding forces and energies, and the first 10 vibrational frequencies for each class. (c) The predicted 3D structures of the generated protein sequences. (d) The fractional content of secondary structure elements.

In this case, we illustrate how SciToolAgent manages a multi-step workflow for protein design. Given a user query (Fig. 3a), the process begins with protein sequence generation, using Chroma [26] to create sequences based on predefined secondary structure types (for instance, α -helix, β -sheet, or mixed). Next, unfolding force and energy calculations are performed using ProteinForceGPT [16] to assess the mechanical stability of the generated sequences. The protein structure is then predicted using ESMFold [27], followed by vibrational frequency computation with Anisotropic Network Model (ANM) [28] implemented in ProDy [29] to evaluate the dynamic stability of the structure. Finally, secondary structure analysis is carried out using the Dictionary of Secondary Structure of Proteins (DSSP) module from BioPython [30] to confirm the structural elements. The results (Fig. 3b-c) from this workflow demonstrate the capability of SciToolAgent to autonomously orchestrate multiple tools and produce reliable outcomes. In contrast, the baseline methods ReAct and Reflexion failed to address this task effectively due to their incorrect or missing tools.

2.3.2 Chemical reactivity prediction

Predicting chemical reactivity is a critical aspect of drug design and organic synthesis. Accurate predictions of how chemical compounds will react with one another can accelerate the development of new compounds and reduce the need for labor-intensive experimental processes. In this case, we employ SciToolAgent to predict the reactivity of amide condensation reactions, a crucial class of reactions in organic chemistry. Specifically, we aim to predict the reactivity of various chemical substrates based on molecular features and identify the most effective machine learning algorithms for this task. The dataset (Fig. 4c) used for this study comprises different combinations of reactants along with the corresponding reaction products, and the conversion rate of the product, quantified by the liquid chromatography area percent (LCAP) at UV 254 nm. The goal is to classify the reactivity of these substrate combinations into three categories: low, medium, and high activity. The workflow (Fig. 4a) begins with an initial query prompting SciToolAgent to predict reactivity using various molecular features, such as RDK fingerprints, Morgan fingerprints, and electrical descriptors, and to test a machine learning algorithm, like the Multi-Layer Perceptron (MLP), to determine which feature provides the best predictive performance. SciToolAgent plans the steps accordingly, first generating each type of fingerprint and descriptor, then training and testing the MLP classifier on each feature set. The results (Fig. 4d) indicate that electrical descriptors outperform the other features in predicting reactivity. Based on this finding, a follow-up query asks SciToolAgent to further explore the performance of different machine learning algorithms, specifically comparing MLP, AdaBoost, and Random Forest classifiers, using only electrical descriptors as input features (Fig. 4b). SciToolAgent then generates a new plan to test each algorithm sequentially on this feature set, evaluating their predictive accuracy in classifying reactivity. By comparing the outcomes, SciToolAgent determines which algorithm is most effective with the electrical descriptors. The results (Fig. 4e) reveal that the Random Forest algorithm yields the highest prediction accuracy among the tested algorithms, making it the optimal choice for this task. Notably, ReAct encountered tool redundancy and Reflexion encountered hallucinations in completing this task.

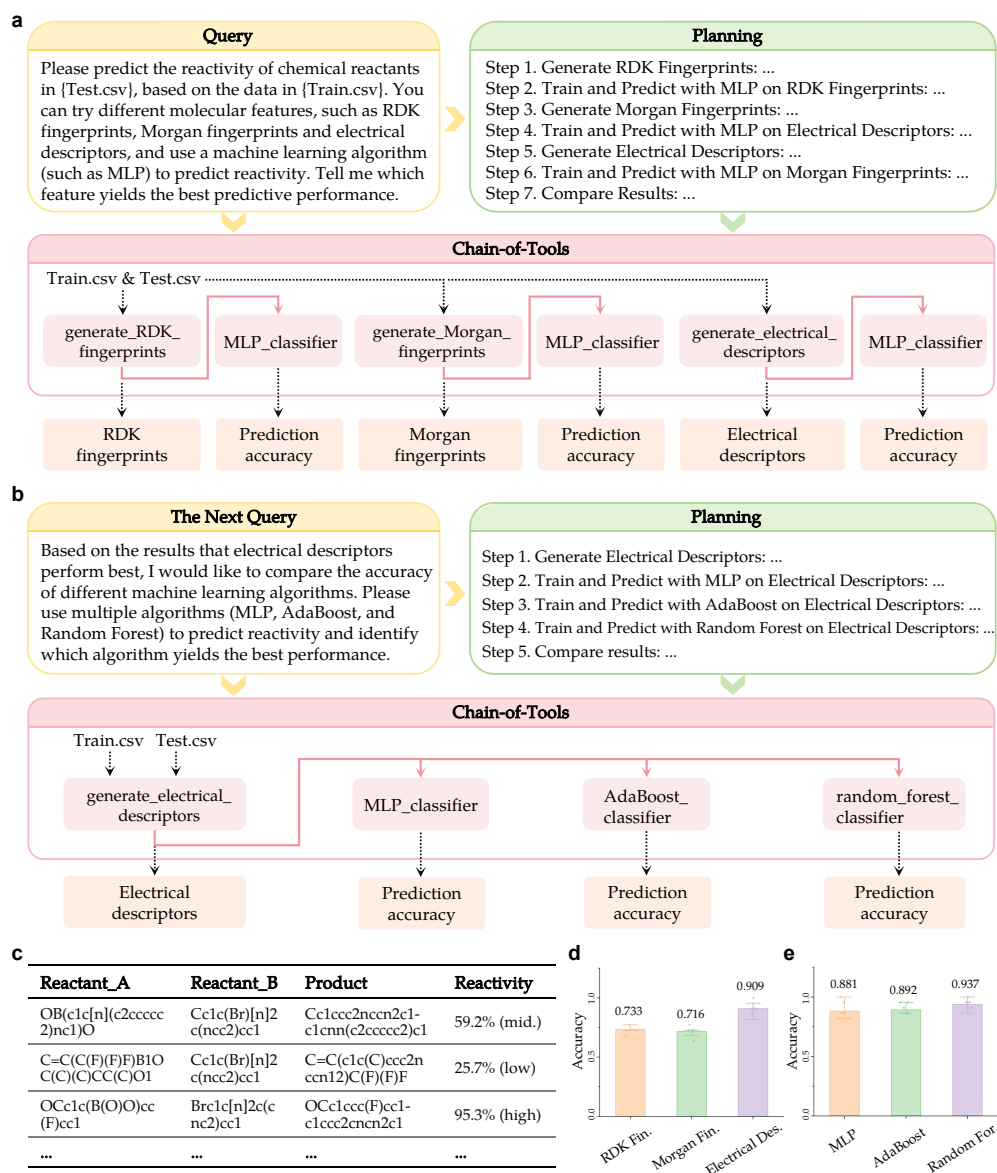


Fig. 4 Workflow and results of machine learning-based chemical reactivity prediction using SciToolAgent. (a) and (b) Overview of the input query, tool planning, chain-of-tools, and corresponding outcomes. (c) The dataset used for training and testing. (d) The prediction accuracy with different molecular features. (e) The prediction accuracy with different machine learning algorithms. Data are presented as mean values. The number of experimental replicates is $n=8$, and the error bars are standard error of the mean.

2.3.3 Chemical synthesis and analysis

Chemical synthesis is a cornerstone of chemistry, pivotal for the design and creation of novel compounds, particularly in pharmaceutical research. Accurately predicting reaction outcomes and understanding the properties of synthesized molecules are crucial steps for efficient and safe drug design. In this case, we demonstrate how SciToolAgent automates a synthesis and analysis pipeline, encompassing reaction prediction, product characterization, intellectual property assessment, and safety evaluation.

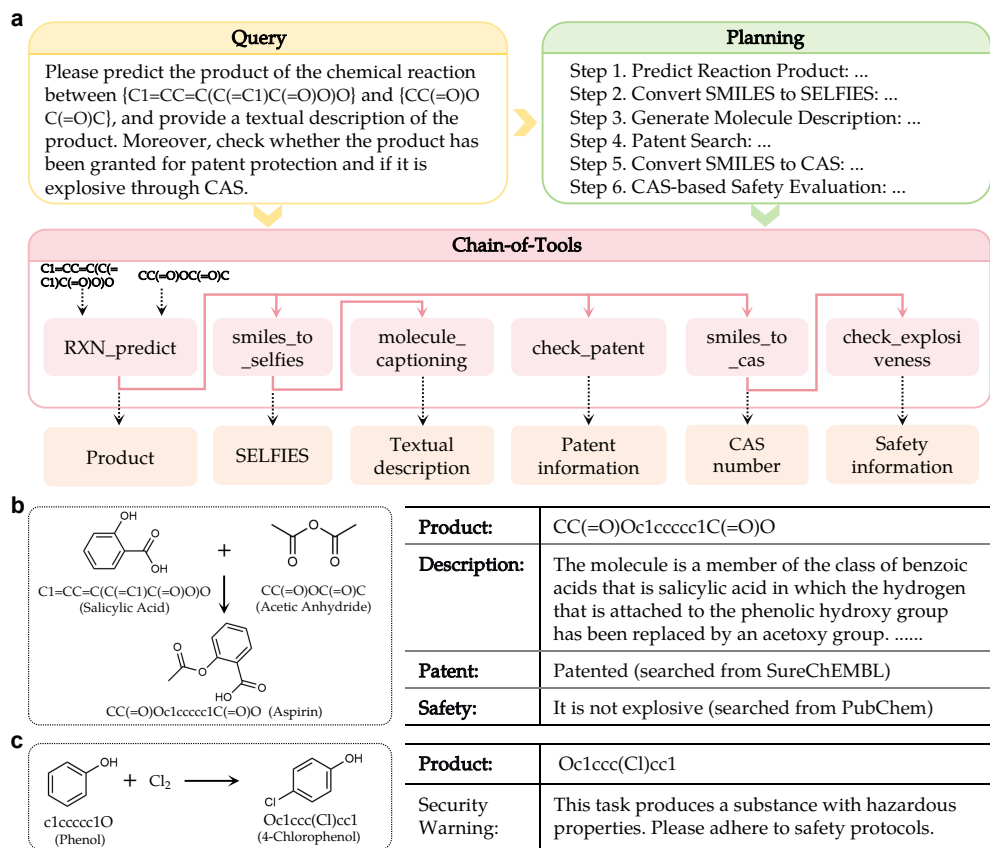


Fig. 5 Workflow and results of chemical synthesis and analysis using SciToolAgent. (a) Overview of the input query, tool planning, chain-of-tools, and corresponding outcomes. (b) Results of the chemical reaction between Salicylic Acid and Acetic Anhydride, including the product and the corresponding description, patent, and safety information. (c) Results of the chemical reaction between Phenol and Chlorine, issuing a security warning due to the product's toxicity.

The first task involves predicting and analyzing the outcome of an esterification reaction between salicylic acid and acetic anhydride. SciToolAgent autonomously plans and executes this multi-step process by orchestrating the necessary tools in sequence,

as shown in Fig. 5a. Specifically, SciToolAgent first employs the RXN-for-chemistry [31] tool to predict the reaction product (denoted by SMILES). Following this, the agent converts the SMILES notation of the product into the SELFIES format for compatibility and uses a molecule captioning tool (BioT5+ [32]) to generate a textual description. Next, the agent conducts a patent search using a patent check tool across the SureChEMBL database [33]. Finally, for explosiveness assessment, the agent converts the SMILES notation to a CAS number and retrieves relevant safety information from PubChem [34]. The obtained results are summarized in Fig. 5b.

Additionally, we conducted a similar query for another reaction: the chlorination of phenol, where phenol reacts with chlorine (Fig. 5c). SciToolAgent follows a comparable workflow for this query, beginning with reaction prediction. However, the internal safety check module in SciToolAgent identifies the product (4-chlorophenol) as a hazardous compound, and issues a security warning indicating that the resulting product is toxic and requires careful handling and specific safety precautions. This built-in safety check is crucial, as it alerts researchers to potential risks early in the workflow, helping to prevent accidental exposure to toxic substances and ensuring that safety protocols are adhered to throughout the experimental process. Conversely, neither ReAct nor Reflexion incorporated safety checks during the synthesis process, which led to substantial risks in chemical experimentation. While both methods were able to predict reaction outcomes, they failed to identify the toxic nature of the product in the chlorination of phenol case.

2.3.4 MOF materials screening

Metal-Organic Frameworks (MOFs) are a class of crystalline porous materials with versatile applications in areas such as gas storage, catalysis, drug delivery, and separation processes. The identification of optimal MOFs for specific scenarios often involves screening materials based on multiple performance criteria, such as high thermal stability and efficient adsorption capacity. In this case, we demonstrate how SciToolAgent can streamline the MOF screening process by automating the selection and analysis of MOFs based on predefined criteria.

Given a batch of MOFs from MOFXDB [35], the workflow (Fig. 6a) begins with a user query that prompts SciToolAgent to find MOFs with thermal stability above 400°C, CO₂ adsorption capacity over 100 mg/g, and a price under ¥100. In response, SciToolAgent generates a plan and executes a chain of tools to filter MOFs based on these criteria. Specifically, the agent first assesses the thermal stability of each MOF by using a machine learning-based predictive model (MOFSimplify [36]) that calculates thermal stability scores. Next, the agent predicts their CO₂ adsorption capacities using a molecular simulation software (RASPA2 [37]). Last, the agent inquires about their pricing information. As the pricing retrieval tool requires CAS numbers as input, it first converts the structural data of MOFs (provided as CIF files) into SMILES format and subsequently translates these SMILES strings into CAS numbers. With the obtained CAS numbers, SciToolAgent accesses the commercial chemical database to retrieve market price information.

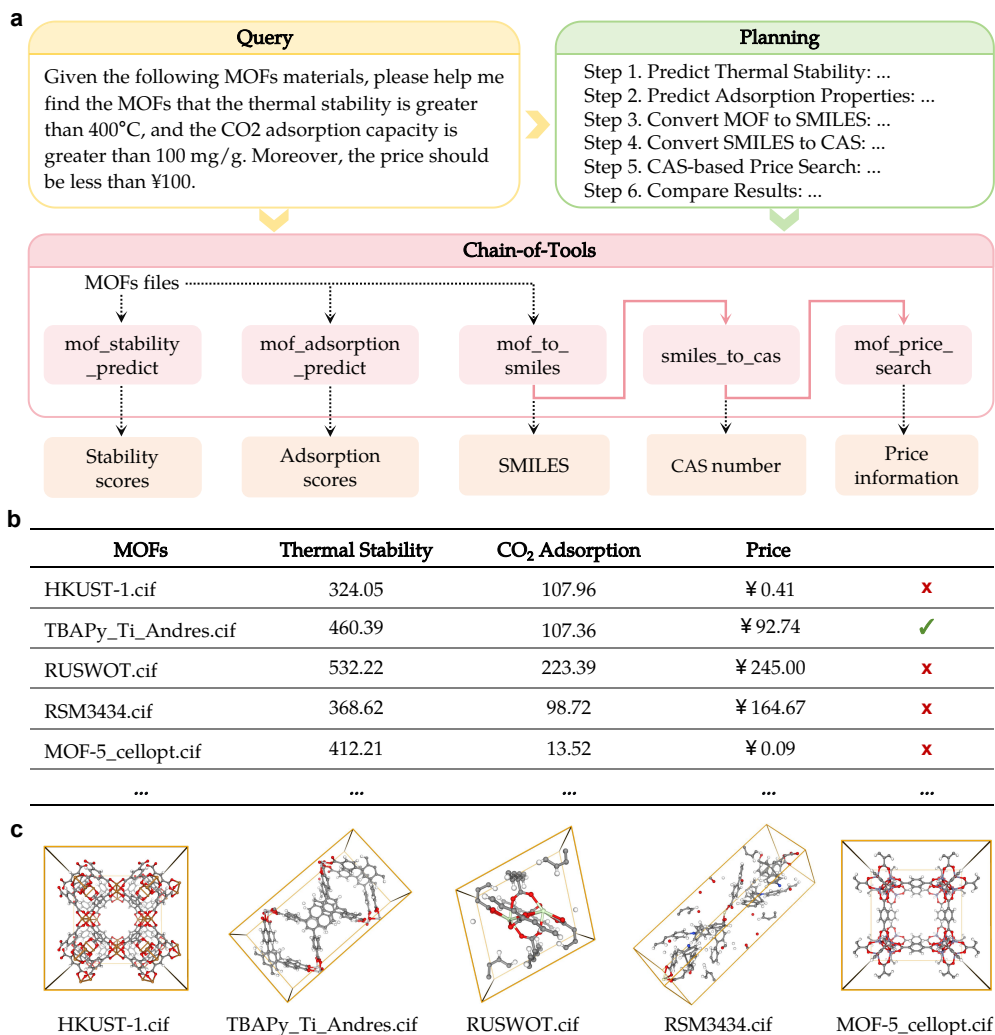


Fig. 6 Workflow and results of MOF materials screening using SciToolAgent. (a) Overview of the input query, tool planning, chain-of-tools, and corresponding outcomes. (b) Summary of the obtained results showing the thermal stability, CO₂ adsorption capacity, and market price for each MOF. The green checkmarks indicate MOFs that meet all specified criteria. (c) Visualization of these MOF structures.

Fig. 6b summarizes the partial results obtained from this automated screening workflow. For instance, by filtering MOFs based on thermal stability, CO₂ adsorption capacity, and price, SciToolAgent successfully identified TBAPy_Ti_Andres.cif as a highly promising candidate that met all predefined criteria. The visualization of selected MOF structures (Fig. 6c) further supports the analysis by highlighting structural elements contributing to stability and adsorption properties, offering valuable insights for subsequent experimental validation. In contrast, ReAct encountered input errors and Reflexion encountered hallucinations when attempting to address this task.

3 Discussion

SciToolAgent is an LLM-powered agent designed to address the challenges of integrating extensive scientific tools for advanced research across multiple domains. The primary advantage of SciToolAgent lies in its integration of diverse scientific tools through SciToolKG, which captures the intricacies of tool dependencies, input/output formats, and application contexts. Previous LLM-based frameworks have been restricted by their limited toolsets and simple task-planning strategies. By contrast, SciToolAgent can dynamically create a chain of tools tailored to the specific requirements of each scientific task. This capability allows researchers to delegate repetitive or computationally intensive steps to SciToolAgent, making scientific inquiry more accessible and efficient for both domain experts and non-experts.

One potential limitation of SciToolAgent lies in the manual construction of the SciToolKG knowledge graph. Although SciToolKG captures relationships and dependencies effectively, its scalability is constrained by the effort required to curate and update tool information. Automated approaches, such as using knowledge extraction techniques from scientific literature or metadata from tool documentation, could further enhance the scalability and granularity of SciToolKG. Additionally, expanding the knowledge graph to incorporate emerging tools and resources will be essential as scientific research continues to evolve. To facilitate extensibility, we provide standardized APIs and templates to support third-party tool integration with minimal effort. Future versions will also include GUI-based registration to reduce the barrier for domain experts without programming experience.

Another challenge lies in the reliance on the underlying LLM’s capabilities to perform effectively across varied scientific tasks. While proprietary models such as GPT-4o demonstrate strong performance, they may be financially and technically inaccessible for all researchers, especially in resource-limited settings. Our experiments with open-source models indicate that fine-tuning with domain-specific data can enhance performance and partially bridge the gap between open-source and proprietary alternatives. However, even with fine-tuning, Qwen2.5-7B-FT still lags behind GPT-4o, especially in complex tool planning and multi-step reasoning.

While challenges related to the scalability of SciToolKG and the reliance on proprietary LLM models remain, SciToolAgent offers a robust foundation for automating complex scientific workflows. Future work will focus on automating the knowledge graph’s maintenance, integrating more tools, and enhancing open-source LLM capabilities to further democratize access to advanced scientific research. Ultimately, SciToolAgent showcases the potential of LLM-driven agents to streamline and empower scientific discovery, making sophisticated tools accessible to a broader audience.

4 Methods

This section presents the methodology of SciToolAgent, including the collection of scientific tools, the construction of SciToolKG and SciToolEval based on these tools, and the implementation details of Planner, Executor, Summarizer, as well as the LLMs behind them.

4.1 Collection of Scientific Tools

The collection of scientific tools for SciToolAgent follows a systematic process aimed at assembling a comprehensive, domain-specific, and functionally diverse toolset. Initially, we identified key scientific domains that would benefit most from LLM integration, including biology, chemistry, and materials science. We then curated a list of tools used frequently in these domains. The current toolset in SciToolAgent includes over 500 tools, spanning a wide range of functionalities. For instance, in biology, we included sequence alignment tools like BLAST [38], protein structure prediction models such as ESMFold [27], and gene editing software like CRISPR-related tools. In chemistry, our toolset features molecular dynamics simulations, cheminformatics libraries like RDKit [39], and compound databases such as PubChem [34]. For materials science, we integrated tools for crystal structure prediction, materials property databases, and MOF property predictors such as MOFSimplify [36].

4.2 Construction of SciToolKG

The Scientific Tool Knowledge Graph (SciToolKG) serves as the backbone of SciToolAgent, providing a structured representation of the relationships, dependencies, and operational details of a vast array of scientific tools. Formally, SciToolKG is represented as a directed graph $G = (V, E)$, where V represents a set of entities (nodes) and E represents a set of relations (edges) between these entities. Specifically, the tool nodes represent individual scientific tools, and the attribute nodes represent various tool properties and metadata. The edges capture the semantic relationships between the tool node and the attribute node.

The construction of SciToolKG followed a three-step process: tool characterization, schema development, and graph population. The initial phase involved defining a set of attributes for each tool, including its purpose, specific functionalities, input/output formats, categories, sources, and safety levels. These attributes were mainly derived from tool documentation. Then, a hierarchical schema was developed to model these attributes logically, including nodes representing tools and their properties and edges denoting relationships such as functional dependencies. The final stage involved populating the KG by encoding tool-specific metadata into a structured graph with triplets. In particular, relationships between tools—such as shared input/output formats or sequential task dependencies—were explicitly captured, creating a rich, interconnected graph. To ensure accuracy and usability, the KG underwent iterative refinement through expert validation.

4.3 Construction of SciToolEval

To evaluate SciToolAgent’s performance, we constructed a comprehensive Scientific Tool Evaluation (SciToolEval) dataset that encompasses a diverse array of real-world scientific tasks. This dataset serves as the first testing ground that can quantitatively assess agents’ capabilities across various scientific domains. The process of constructing SciToolEval involves the following steps.

(1) Tool selection and question generation: The first step is selecting the appropriate tools and generating meaningful questions. For single tool calling scenarios, we include a detailed description of the tool in the prompt and instruct GPT-4o to generate relevant questions. This ensures that the generated questions are directly actionable and pertinent to the tool’s functionality.

(2) Tool execution and answer generation: Once the questions are generated, we proceed to tool execution and answer generation by utilizing the ReACT framework in conjunction with GPT-4o. ReACT [21] has demonstrated its effectiveness in handling a limited number of tools. Therefore, only the tools relevant to the question are fed to ReACT, improving the accuracy of tool invocation.

(3) Manual review: The final step involves manual review to ensure the quality of generated answers. We invite three domain-specific experts to meticulously review the questions, tool usage, and answers to verify their correctness and significance.

Finally, SciToolEval consists of 531 questions associated with the required tools and their corresponding answers. These questions cover single and multiple tools for problem-solving, and they are categorized into different levels based on difficulty: **Level-1** involves simple information queries using a single tool; **Level-2** includes sequential and parallel use of multiple tools, requiring basic planning, reasoning, and summarization skills.

4.4 Implementation of SciToolAgent

SciToolAgent’s architecture revolves around three core components powered by LLMs: Planner, Executor, and Summarizer (Fig. 1c). Each component plays a distinct role in the problem-solving workflow by leveraging the strengths of LLMs.

4.4.1 Planner

The Planner is responsible for devising a strategy to solve the given query. It leverages LLMs to interpret the question, query the SciToolKG, and generate a plan of chain-of-tools. Specifically, a retrieve-augmented generation approach based on the SciToolKG is proposed to identify and sequence the appropriate tools needed. The process begins with the LLM querying the SciToolKG based on the input question, followed by the steps described below.

(1) Full-graph retrieval: We first retrieve the most relevant k tools for the given question in the full graph of SciToolKG. This is achieved by calculating the semantic similarity between the question and all of the triples involved in tool functions within the SciToolKG. The similarity score helps in identifying tools that are most likely to be useful for the given query. In mathematical, let T_i represent a tool in the SciToolKG \mathcal{G} and $S(q, T_i)$ denote the semantic similarity score between the question q and the tool T_i . This retrieval process can be formulated as:

$$\mathbb{T}_{\text{full}} = \text{top-}k\{T_i \mid S(q, T_i), \forall T_i \in \mathcal{G}\}. \quad (1)$$

(2) Sub-graph exploration: Once the initial set of tools is retrieved, we explore all sub-graphs (that is, d -hop neighborhoods) of these tools in SciToolKG. The purpose

of this step is to identify any additional tools that might be required in conjunction with the initially retrieved tools to solve the query. Mathematically, for each retrieved tool T_i , we calculate the semantic similarity between q and the tools $T_i \oplus T_j$, where \oplus denotes semantic concatenation that combines the textual metadata of T_i and T_j . Here, T_j is the tool in the d -hop neighborhoods of T_i . The top- k tools with the highest similarity are obtained by:

$$\mathbb{T}_{\text{sub}} = \text{top-}k\{T_j \mid S(q, T_i \oplus T_j), \forall T_j \in \mathcal{G}_{\text{sub}}^{T_i}\}. \quad (2)$$

(3) Tool combination and ranking: By employing full-graph retrieval and sub-graph exploration for each query, we obtain a total of k^2 tools. To optimize the utilization of these tools, we sort them according to the combined similarity score $S' = S(q, T_i) \times S(q, T_i \oplus T_j)$, prioritizing tools that are not only relevant to the question but also complementary to each other. We then select the top- n tools ($n \leq k^2$) based on the combined similarity:

$$\mathbb{T}_{\text{comb}} = \text{top-}n\{T_i \mid S(q, T_i) \times S(q, T_i \oplus T_j), \forall T_j \in \mathcal{G}_{\text{sub}}^{T_i}, \forall T_i \in \mathcal{G}\}. \quad (3)$$

(4) Chain-of-tools generation: Based on the selected tools and the neighborhood information from SciToolKG, we prompt LLMs to generate a chain-of-tools that outlines the tools required to solve the query. This chain is optimized to ensure that the tools are used in the most effective order, leveraging their dependencies and functionalities. The final chain-of-tools can be represented as:

$$\mathbb{T}_{\text{chain}} = \{T_1 \rightarrow T_2 \rightarrow \dots \rightarrow T_m\} \in \mathbb{T}_{\text{comb}}. \quad (4)$$

Unless otherwise specified, the parameters for retrieval are set as follows: $d = 3$, $k = 5$, and $n = 10$. For measuring semantic similarity S , we calculate the cosine distance between two text embedding vectors obtained from a pretrained embedding model.

4.4.2 Executor

The Executor aims to ensure the effective implementation of the planned chain-of-tools. This component is designed to handle tool inputs, manage execution processes, and address any errors that may arise during execution. It also integrates a robust safety module to monitor and control potentially hazardous inputs or outputs. The execution process involves the four steps as follows.

(1) Input preparation: The Executor uses LLMs to extract the required input parameters for the current tool in the chain. This involves parsing the question, understanding the context, and identifying the necessary data to feed into the tool. Inputs are formatted according to the specifications of the tool recorded in SciToolKG. This may include converting data types, structuring the input in a specific format, or integrating multiple inputs.

(2) Tool execution: The Executor invokes the tool with the prepared inputs, which involves interfacing with the tool’s API. The execution process is actively monitored to track progress, capture outputs, and detect any anomalies or errors. The

Executor logs execution details to ensure traceability. Once the tool has been executed, the outputs are captured, and further processed as needed to be compatible with the subsequent tools in the chain.

(3) Error handling and retries: The Executor is equipped with protocols to detect errors during tool execution. Errors can occur due to incorrect inputs, tool malfunctions, or other unforeseen issues. When an error is detected, the Executor adjusts the inputs based on predefined rules or heuristics to rectify the issue.

(4) Retrieve-based safety check module: In parallel with the execution process, the Executor incorporates a retrieve-based safety check module. This module evaluates each step’s inputs and outputs against a set of predefined safety criteria to prevent the generation of harmful substances or unethical use of scientific tools. To this end, we first construct a comprehensive safeguard database that includes hazardous compounds collected from PubChem and toxic proteins collected from UniProtKB. By comparing the outcomes with the substances contained in this database, we can assess the potential harmfulness of molecules or protein sequences generated during execution.

Specifically, given a molecule x , the safety check module calculates the feature similarity between x and all entries in the safeguard database D . The similarity is quantified using the average of Tanimoto coefficient, Dice coefficient and Cosine coefficient, which measure the degree of similarity between molecular fingerprints [40]:

$$\hat{S}_{\text{mol}}(x, D) = \max_{y_i \in D} \frac{1}{3} (\text{Tanimoto}(x, y_i) + \text{Dice}(x, y_i) + \text{Cosine}(x, y_i)). \quad (5)$$

For proteins, we use the Smith-Waterman algorithm [41] for pairwise local alignment between the given sequence and entries in the safeguard database, and the percentage of matching serves as the similarity score:

$$\hat{S}_{\text{prot}}(x, D) = \max_{y_i \in D} \text{Smith-Waterman}(x, y_i). \quad (6)$$

If $\hat{S}(x, D)$ exceeds a certain threshold $\delta = 0.95$, indicating a close to known hazardous or toxic entities, the safety module flags this output as potentially dangerous. To ensure the efficiency of execution, only the tools marked with high risks in SciToolKG are required to pass through the safety check module.

4.4.3 Summarizer

After execution, the Summarizer compiles and synthesizes outputs from various tools to generate the final answer, ensuring coherence and accuracy. It also assesses the success of the problem-solving process and, if necessary, prompts the Planner to refine the chain-of-tools for optimal results. The implementation of the Summarizer involves several key steps.

(1) Synthesis of outputs: The Summarizer collects the outputs from all the tools executed in the chain and integrates them into a cohesive response. This involves 1) combining the outputs from different tools and ensuring that all relevant information is included, 2) verifying that the outputs from different tools align with each other

and do not contain conflicting information, and 3) structuring the final answer in a logical and easy-to-understand format.

(2) Iterative refinement: If the initial chain-of-tools fails to produce satisfactory results, as determined by the Summarizer, it prompts the Planner to refine the plan. This refinement involves identifying reasons for any failures or suboptimal outcomes, suggesting modifications to the chain-of-tools (such as adding, removing, reordering, or retrieving tools), and repeating the execution process with the refined chain-of-tools to achieve improved results.

4.4.4 Foundation models

The foundation model behind the Planner, Executor and Summarizer in SciToolAgent are large language models (LLMs). Our experiments have tried different LLMs, including the well-known proprietary model - OpenAI’s GPT-4o, as well as the state-of-the-art open-source models - Qwen2.5-72B. Moreover, we fine-tuned Qwen2.5-7B, an open-source LLM with a smaller parameter scale but higher efficiency, using specific instructions for tool learning and deployed it locally on a GPU server.

(1) Instruction generation: To enhance the performance of Qwen2.5-7B, we generate specific instructions tailored to scientific tool learning. These instructions guide the model in understanding the functionalities of various tools and their appropriate usage. The instruction generation process is similar to the construction of the SciToolEval dataset, utilizing SciToolKG and GPT-4o for automated generation (encompassing tool selection, question generation, tool execution, and answer generation). This approach yields a large number of instructions, although they are not manually reviewed.

(2) Fine-tuning LLMs with instructions: We fine-tuned Qwen2.5-7B using LoRA (Low-Rank Adaptation [42]), leveraging the constructed instruction dataset to enhance the model’s capacity for tool-specific planning and execution. Separate fine-tuning was conducted for the Planner, Executor, and Summarizer, with task-specific instructions ensuring the optimization of each component’s functionality. The training loss is measured using the next token prediction loss [43], defined as:

$$L = - \sum_{t=1}^T \log P(y_t | y_{<t}, X; \theta), \quad (7)$$

where y_t represents the target token at time step t , X is the input text, and θ denotes the model parameters. This fine-tuning process significantly enhances the instruction following capability of Qwen2.5-7B, making it a cost-effective choice for implementing the Planner, Executor and Summarizer.

Data Availability

The toxic compound data were obtained from PubChem (<https://pubchem.ncbi.nlm.nih.gov/#query=toxic&tab=compound>), and the toxic protein data were sourced from UniProtKB (<https://www.uniprot.org/uniprotkb?query=toxic&facets=reviewed%3Atrue>).

Code Availability

The source code of SciToolAgent can be found at GitHub (<https://github.com/hicai-zju/scitoolagent>).

References

- [1] Birhane, A., Kasirzadeh, A., Leslie, D. & Wachter, S. Science in the age of large language models. *Nature Reviews Physics* **5**, 277–280 (2023).
- [2] Schick, T. *et al.* Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems* **36** (2024).
- [3] Yang, R. *et al.* GPT4Tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems* **36** (2024).
- [4] Guo, T. *et al.* What can large language models do in chemistry? A comprehensive benchmark on eight tasks. *Advances in Neural Information Processing Systems* **36**, 59662–59688 (2023).
- [5] Zhao, W. X. *et al.* A survey of large language models. *arXiv:2303.18223* (2023).
- [6] Min, B. *et al.* Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys* **56**, 1–40 (2023).
- [7] Wang, L. *et al.* A survey on large language model based autonomous agents. *Frontiers of Computer Science* **18**, 186345 (2024).
- [8] Ramos, M. C., Collison, C. J. & White, A. D. A review of large language models and autonomous agents in chemistry. *arXiv:2407.01603* (2024).
- [9] Boiko, D. A., MacKnight, R., Kline, B. & Gomes, G. Autonomous chemical research with large language models. *Nature* **624**, 570–578 (2023).
- [10] Janakarajan, N., Erdmann, T., Swaminathan, S., Laino, T. & Born, J. Language models in molecular discovery. *arXiv:2309.16235* (2023).
- [11] Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari & Andrew D White Augmenting large language models with chemistry tools. *Nature Machine Intelligence* **6**, 525–535 (2024).
- [12] McNaughton, A. D. *et al.* CACTUS: Chemistry agent connecting tool usage to science. *ACS Omega* **9**, 46563–46573 (2024).
- [13] Jin, Q., Yang, Y., Chen, Q. & Lu, Z. GeneGPT: Augmenting large language models with domain tools for improved access to biomedical information. *Bioinformatics* **40**, btae075 (2024).
- [14] Huang, K. *et al.* CRISPR-GPT: An LLM agent for automated design of gene-editing experiments. *arXiv:2404.18021* (2024).

- [15] Liu, H. & Wang, H. GenoTEX: A benchmark for evaluating LLM-based exploration of gene expression data in alignment with bioinformaticians. *arXiv:2406.15341* (2024).
- [16] Ghafarollahi, A. & Buehler, M. J. ProtAgents: Protein discovery via large language model multi-agent collaborations combining physics and machine learning. *arXiv:2402.04268* (2024).
- [17] Jia, S., Zhang, C. & Fung, V. LLMatDesign: Autonomous materials discovery with large language models. *arXiv:2406.13163* (2024).
- [18] Kang, Y. & Kim, J. ChatMOF: an artificial intelligence system for predicting and generating metal-organic frameworks using large language models. *Nature Communications* **15**, 4705 (2024).
- [19] Wu, H. *et al.* ChatEDA: A large language model powered autonomous agent for EDA. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **43**, 3184–3197 (2024).
- [20] Ni, B. & Buehler, M. J. MechAgents: Large language model multi-agent collaborations can solve mechanics problems, generate new data, and integrate knowledge. *Extreme Mechanics Letters* **67**, 102131 (2024).
- [21] Yao, S. *et al.* ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations* (2023).
- [22] He, J. *et al.* Control risk for potential misuse of artificial intelligence in science. *arXiv:2312.06632* (2023).
- [23] Liu, X. *et al.* ToolNet: Connecting large language models with massive tools via tool graph. *arXiv:2403.00839* (2024).
- [24] Hao, S., Liu, T., Wang, Z. & Hu, Z. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. *Advances in neural information processing systems* **36** (2024).
- [25] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. & Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems* **36** (2024).
- [26] Ingraham, J. B. *et al.* Illuminating protein space with a programmable generative model. *Nature* **623**, 1070–1078 (2023).
- [27] Lin, Z. *et al.* Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
- [28] Atilgan, A. R. *et al.* Anisotropy of fluctuation dynamics of proteins with an elastic network model. *Biophysical journal* **80**, 505–515 (2001).
- [29] Bakan, A., Meireles, L. M. & Bahar, I. Prody: protein dynamics inferred from theory and experiments. *Bioinformatics* **27**, 1575–1577 (2011).
- [30] Cock, P. J. *et al.* Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25**, 1422 (2009).
- [31] Schwaller, P. *et al.* Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. *ACS central science* **5**, 1572–1583 (2019).
- [32] Pei, Q. *et al.* BioT5+: Towards generalized biological understanding with IUPAC integration and multi-task tuning. *Findings of the Association for Computational Linguistics* 1216–1240 (2024).
- [33] Papadatos, G. *et al.* SureChEMBL: a large-scale, chemically annotated patent

- document database. *Nucleic Acids Research* **44**, D1220–D1228 (2016).
- [34] Kim, S. *et al.* Pubchem substance and compound databases. *Nucleic acids research* **44**, D1202–D1213 (2016).
 - [35] Bobbitt, N. S. *et al.* MOFX-DB: An online database of computational adsorption data for nanoporous materials. *Journal of Chemical & Engineering Data* **68**, 483–498 (2023).
 - [36] Nandy, A. *et al.* Mofsimply, machine learning models with extracted stability data of three thousand metal–organic frameworks. *Scientific Data* **9**, 74 (2022).
 - [37] Dubbeldam, D., Calero, S., Ellis, D. E. & Snurr, R. Q. RASPA: molecular simulation software for adsorption and diffusion in flexible nanoporous materials. *Molecular Simulation* **42**, 81–101 (2016).
 - [38] BLAST: Basic local alignment search tool. URL <https://blast.ncbi.nlm.nih.gov>. Accessed 2 Oct. 2024.
 - [39] RDKit: Open-source cheminformatics software. URL <http://www.rdkit.org>. Accessed 6 Oct. 2024.
 - [40] Bajusz, D., Rácz, A. & Héberger, K. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics* **7**, 1–13 (2015).
 - [41] Smith, T. F., Waterman, M. S. *et al.* Identification of common molecular subsequences. *Journal of molecular biology* **147**, 195–197 (1981).
 - [42] Hu, E. J. *et al.* LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations* (2022).
 - [43] Vaswani, A. *et al.* Attention is all you need. *Advances in Neural Information Processing Systems* **30** (2017).